M2M Protocols for Constrained Environments in the Context of IoT: A Comparison of Approaches

Edielson P. Frigieri, Daniel Mazzer and Luís F. C. G. Parreira

***Abstract*— The Internet of Things movement opens new possibilities for services and business along with new technological challenges, such as power efficiency, operation in constrained environments, security, and privacy. With the expectation of a high amount of devices connected in this Future Internet, scalability is also assumed to be a challenge. To address these limitations, several protocols are being proposed. In this paper, two of them, MQTT and COAP, are presented and qualitatively compared, summarizing their main features and limitations, highlighting the best scenarios where each approach is more suitable.**

Keywords*— *Constrained, Internet of Things, Machine-to-Machine, Security, Scalability.

I. INTRODUÇÃO

The possibility of connecting people and smart objects through a common infrastructure – the Internet, has been the focus of many recent researches on Information and Communications Technologies (ICT). This new approach is named Internet of Things (IoT) and the main idea refers to an unified network for interconnecting people and any kind of *thing*, which can be a real or virtual objects, e.g. a hardware device or a web service [1].

The IoT scenario uses the Internet for conducting information exchange and communication aiming at achieving different kinds of services, like monitoring, tracking, positioning, and smart recognitions. Along with the new possibilities appear new challenges like power efficiency, operability in constrained environments (devices, bandwidths, networks, etc.) and concerns about security and privacy [2]. If this new technology doesn't guarantee the safety of private information, users will be averse to adopt it to their environment and life [3].

Since this technology shift is expected to be greater than the one caused by the advent of mobile phones, serious scalability problems can be highlighted in the context of standardized machine-to-machine (M2M) protocols while facilitating human-machine interaction [4]. Many protocols have been designed for unconstrained environments, where the number of devices is limited to hundreds or thousands. However, IoT scenarios present very constrained environments with millions or even billions of devices which are motivating the design of new protocols, focused on networking/computing constrained environments which demand only a few transfers of bytes per day and should run on battery powered devices for years.

In order to choose the best protocol, a careful analysis should be done on the target application, as well as on the protocol features and requirements.

Following these tendencies, this work aims to present, analyze qualitatively and discuss some of the M2M proposed protocols, identifying their main features and limitations and highlighting the best scenarios where each one can be applied.

For this aim, the remaining of this paper is organized as follows: Section II presents some M2M protocols for Internet of Things scenarios; Section III discusses them, summarizing their main features and limitations; finally, the Section IV draws the paper conclusions.

II. M2M PPROTOCOLS FOR INTERNET OF THINGS

There are several protocols proposed for M2M communication with focus on constrained environments. The great majority of IoT and M2M protocols are IP based, making use of already available networks like Wi-Fi, Ethernet, 6LoWPAN and mobile. Among them, MQTT (*Message Queue Telemetry Transport*) [5] and CoAP (*Constrained Application Protocol)* [6] are frequently adopted. They have been analyzed for different perspectives [7] and are popular among commercial products with high availability of commercial web services, and caught the interest of the open source community [8]. The advantage of using open protocol is that the developer may focus on application business, leaving the message delivery on charge of the protocol. Also, these protocols are suitable for use on low memory hardware and low processing power microcontrollers [9].

A. *MQTT*

Created by IBM and Eurotech, the MQTT is an open protocol designed to be simple, lightweight, and easy to implement: suitable features for embedded devices with limited battery, processor and/or memory resources. The small transport overhead (fixed-length header of 2 bytes) makes the MQTT an interesting solution for unreliable networks with restricted resources, such as low bandwidth and high-latency [5]. This protocol is based on a lightweight broker using publish/subscribe message pattern where the broker server acts as an intermediary for messages sent from a publisher client to subscriber clients, providing one-to-many message distribution and decoupling of use case application. All messages addressed to a specific topic, sent by publisher, will be delivered, by the broker server, to the subscribers of this

Edielson Prevato Frigieri, Daniel Mazzer and Luís F. C. G. Parreira¸ Department of Computer Engineering, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí-MG, Brazil, E-mails: edielson@inatel.br, daniel.mazzer@inatel.br, and lfgambogi@gmail.com.

topic[1]. Some important features can be highlighted as: *Keep-Alive* message (PINGREQ, PINGRESP) where broker can detect client disconnection even when it doesn't send explicit DISCONNECT messages; the *Retain* message where a PUBLISH message on a specific topic can be retained in the broker allowing a new connected subscriber, on the same topic, to receive it; *Last Will* message (specified in CONNECT message with topic, QoS, and retain) allows subscribed clients being informed about an unexpected client disconnection; Durable subscription that keeps all subscriptions retained in the broker when a client is disconnected, and allows them to be recovered on client reconnection.

These MQTT features and characteristics driven its application to solutions where low battery consumption is a pre-requirement and where there is low bandwidth available or intermittent connection [10], which characterizes the first layer in a sensor network application.

A new standard version for MQTT called MQTT-SN, with focus on sensor networks, was developed for running in different networks than TCP/IP, like UDP, 6LoWPAN[2], customized serial protocols, and customized radio frequency protocols as the IEEE standard 802.15.4. One of the main differences about the two standards, besides the network layer they are focused on, is the simplification in the messages exchanged among broker and clients, using "pre-defined" topic identifiers and short topic names in addition to small messages format [11]. As MQTT-SN is not fully standardized, different implementations may not be compatible each other.

B. CoAP

The CoAP protocol was designed by the Constrained RESTful Environments (CoRE) Working Group of Internet Engineering Task Force (IETF). Adapted from HTTP, it was optimized for devices with constrained power and processing capabilities usually applied to smart objects in the IoT environment [6]. Running over UDP transport protocol, CoAP specifies a minimal subset of REST requests including POST, GET, PUT, and DELETE, supporting resource caching and built-in resource discovery.

CoAP adopts a *request/response* model, where each device acts as "client" or "server" and the resources can be accessed by URIs. Different from HTTP, the connection is not established before message exchanging. The communication happens in an asynchronous way. There are four types of messages: CON (Confirmable), NON (Non-Confirmable), ACK (Acknowledgment), and RESET.

In comparison to the HTTP, CoAP is more cost-effective because it performs less data exchange between client and server, resulting in lower power consumption when using cheaper equipment in both sides of the connection [12]. In summary, its key characteristics can be outlined as: compact binary header in combination with the UDP based transport that reduces the overhead data and consequently decreases the delay and minimizes battery usage during transmission; support for the asynchronous information push (observe option) that allows smart objects to send resource information only when there is a change. The device can stay in "sleep mode" most of the time, which means a reduction in the power consumption; use of a minimal subset of the REST requests enables the usage of hardware with lower requirements when compared with HTTP.

These characteristics favored CoAP for solutions targeted to embedded devices with severe memory and power supply restrictions in addition to constrained networks. The REST architecture and the easily translation to HTTP enables to create scenarios where old web clients can access CoAP servers transparently using proxies that make a set of CoAP resources available like regular http:// or https:// URIs [13].

III. Comparison of The Presented Protocols

Since the paper focuses on comparing the proposed protocols for constrained environments, the attention in the qualitative analysis is devoted to defining the protocol characteristics that are best fitted in a baseline scenario. As defined by Sen (2010), the principal constraints in WSNs (Wireless Sensor Networks), the base for IoT scenarios, are: energy constraints, limit processing capability, memory limitations, unreliable networks, higher latency in communication, and unattended network operation. According to the related constrained characteristics, this paper adopts a baseline scenario, as presented in Fig. 1 composed by sensors nodes defined as "S", using 8-32 bits microcontrollers with 16-190 MHz clock rate, 8K-64M bytes RAM (DRAR, SRAM) memory, and 64K-1M bytes flash memory [15]. All sensors are powered by batteries and exchange data over an unreliable network that could be wired or wireless. The data transfer between "S" nodes is considered in the range of 15kbps to 1Mbps. The collected data are sent to the "Cloud" computing environment through the *Gateway*, with a minimum data transfer of 50kbps, where they become available to the users in some way.

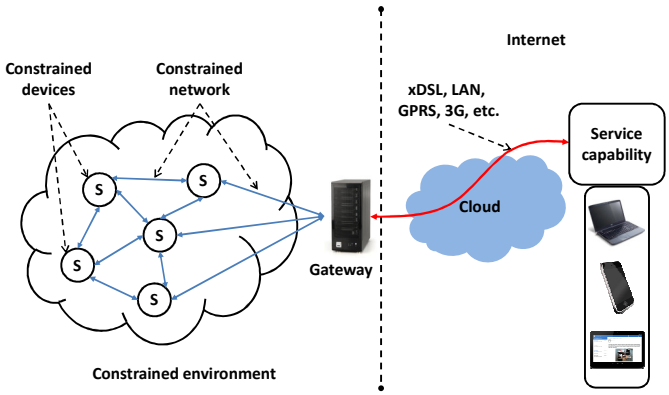

Fig. 1. Baseline scenario with constrained environment for IoT applications.

Some metrics must be defined in order to populate the proposed protocols comparison. For a comprehensive analysis, the following topics will be considered: Implementation (size cost); Data transport (transmission cost); Communication patterns; Reliability and QoS; Scalability; Security and availability of open source implementations.

[1] Every MQTT message includes a topic that classifies it. MQTT brokers use topics to determine which subscribers should receive messages published to the broker.

[2] 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks): encapsulation and header compression mechanisms that allow IPv6 packets to be sent to and received from over IEEE 802.15.4 based networks;

C. Implementation

In terms of implementation, MQTT has the simpler protocol specification [5], therefore facilitating client development. The CoAP clients act as HTTP clients but in binary mode, which becomes simpler than HTTP, but still more complex than MQTT. Based on those characteristics, both MQTT and CoAP are suitable for the "S" nodes implementation where there are energy constraints, limit processing capability and memory limitations with a small advantage to MQTT.

D. Data transport

MQTT employ connection oriented communication given by TCP, which is more costly than UDP used by CoAP protocol. The use of TCP means more data exchanged between client and server. If TCP or UDP is not necessary, one alternative is to select the MQTT-SN over 6LoWPAN or even ZigBee®[3], avoiding the complexity of the entire TCP/IP stack. CoAP was also designed for running over constrained networks, such as 6LoWPAN, with the goal of keeping message overhead small, thus limiting the need for fragmentation what cause significant reduction in packet delivery probability [6].

The message format in both MQTT and CoAP are binary but an interesting point about MQTT is that the entire header has only two bytes, making it an interesting solution for networks with low transmission rate, represented by the connection between "S" nodes in the proposed scenario (Fig. 1). In the case of the payload, MQTT is agnostic and data can be transmitted without specific type or format whereas CoAP works with binary payload.

E. Communication patterns

Before performs the comparison, it is necessary to introduce the IoT communication patterns that can be defined as *Telemetry*, *Inquiries*, *Commands*, and *Notifications*. In *Telemetry*, information flows from devices to the cloud informing status changes in the device. Fig. 2 presents an example of *Telemetry* communication pattern for both protocols. According to Fig. 2 (b), *Telemetry* pattern is not suitable for CoAP because the connection must be started from system (client) to the device (server), which can confront addressing problems like mobile roaming or NAT. The MQTT publish/subscribe model matches with *Telemetry* pattern facilitating its application.

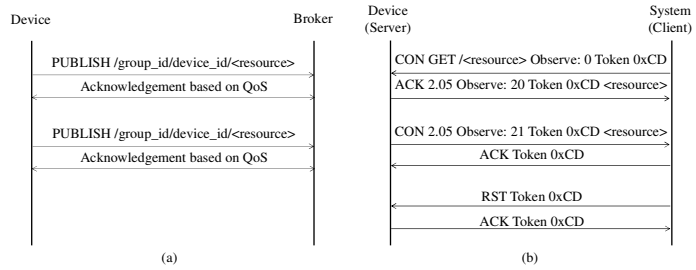

Fig. 2. *Telemetry* communication pattern example for (a) MQTT, (b) CoAP.

For the *Inquiries* pattern, requests come from devices to the cloud for collecting required information, as illustrated in Fig. 3. According to Fig. 3 (b), CoAP have better performance for this pattern since they are based on request/response model. When using MQTT for *Inquiries* pattern, there is the necessity of defining a response topic for communication since there is not a built in response path support which configures an implementation difficulty.

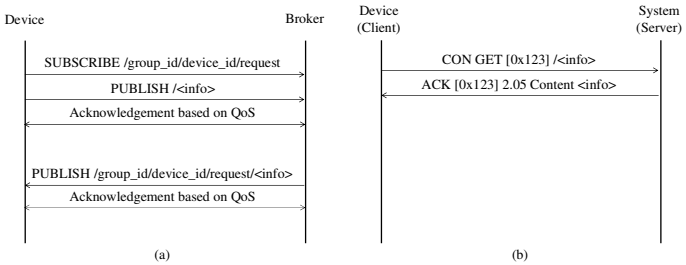

Fig. 3. *Inquiries* communication pattern example for (a) MQTT, (b) CoAP.

In *Commands* pattern, commands are sent from systems to device/devices for performing specific activities. Fig. 4 presented the examples for both protocols. Analyzing this scenario, CoAP presents, for this pattern, the same addressing problems as detailed in *Telemetry*. In the case of MQTT, there is no built in result path support, which requires the definition of a result topic for working as an answer path. Also, old commands can be delivered when using "retain" flag, or new commands can be lost if not using it.

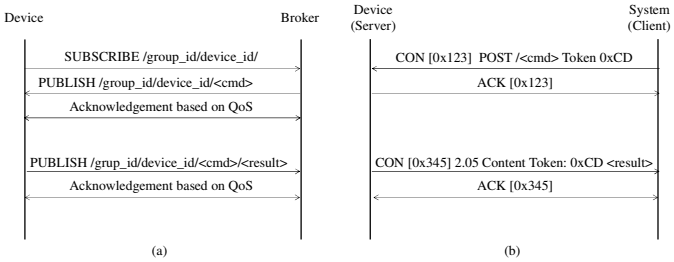

Fig. 4. *Commands* communication pattern example for (a) MQTT, (b) CoAP.

Finally, in *Notifications* the information flows from systems to device/devices handling status changes in the physical world, as presented in Fig. 5. In this pattern, the CoAP addressing problems is also present, On the other hand, MQTT publish/subscribe model fits in the notification architecture having problems only if a better flow control is required for big amount of data at high data rates.

F. Reliability and QoS

All communication patterns can increase their reliability using some level of QoS, which guarantees data delivery or even avoids duplication of packets. Both presented protocols have QoS options that can be used depending on the desired data flow control. MQTT V3.1 supports 3 levels of Quality of Service (QoS) that represents the message delivery confidence [5]. Fig. 6 shows the packet exchange according to this 3 different QoS levels. In Fig. 6 (a), the QoS level 0 is employed and the publisher sends a message at most once and does not check if the message arrived to its destination. This lower level is also called "fire and forget" and the message can be lost depending on the network condition.

[3] ZigBee®: specification for a suite of high-level communication protocols used to create personal area networks built from small, low-power digital radios. ZigBee is based on an IEEE 802.15.4 standard;

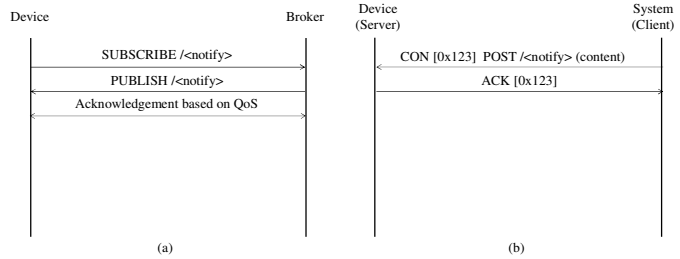

Fig. 5. *Notifications* communication pattern example for (a) MQTT, (b) CoAP,

The QoS level 1, also called "acknowledged delivery", is illustrated on Fig. 6 (b). The publisher sends the message at least once and checks the delivery status using the PUBACK status check message. However, if PUBACK is lost, the broker server can probably send the same message twice, since it has no confirmation of the message being delivered. In QoS Level 2, also called "assured delivery" and shown in Fig. 6 (c), the messages are delivered exactly once using a 4-way handshake. Due to its complicated process, it is possible to have relatively longer end-to-end delays, but there is no messages loss in this level.

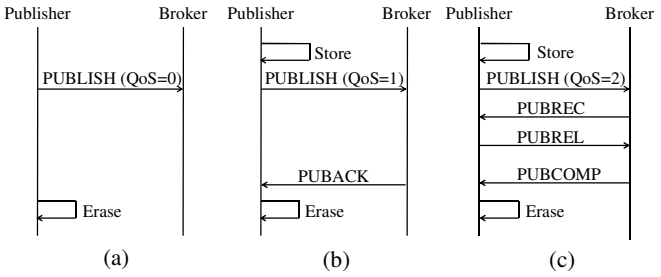

Fig. 6. Publish/subscribe messaging model with different QoS level.

In the case of CoAP protocol, the CON and NON messages acts as QoS levels, where the CON message represents the higher QoS level, as exemplified in Fig. 7 (a) and (b). A request sent using NON type has no acknowledge message sent back by the receiver, which characterizes a low QoS level. This type of exchange is illustrated in Fig. 7 (c). The higher is the QoS level, the greater is the exchange of packets. If messages loss is not such a problem, a lower QoS level can be used resulting in lower bandwidth and lower end-to-end delay [16], which configure wired or wireless constrained networks.

G. Scalability

Architectures based on MQTT protocol can easily scale horizontally because they are based on publish/subscribe model. The strength of this model is based on decoupling in time where publishers and subscribers do not need to be transmitting at the same time. Beyond that, publishers and subscribers don't need to know about each other, which represent a decoupling in space. Events can be produced or consumed in an asynchronous way allowing greater scalability and flexibility [17]. As both protocols rely on broker to exchange messages, the system infrastructure can be easily scaled up if more bandwidth or processing power is needed. CoAP based architectures can also be scalable but in a different mode since devices are considered resources. But, if the observe option provided by CoAP is used in a *Telemetry* interaction model, clients are allowed to monitor the events registering its interest by means of an extended GET request sent to the server node. The server notifies each client node that has an observation relationship with the event. Although, the server acts as a broker and high scalability and efficiency can be performed using caches and intermediaries (proxy) nodes that multiplex the interest of multiple clients (subscribers) in the same event into a single association.

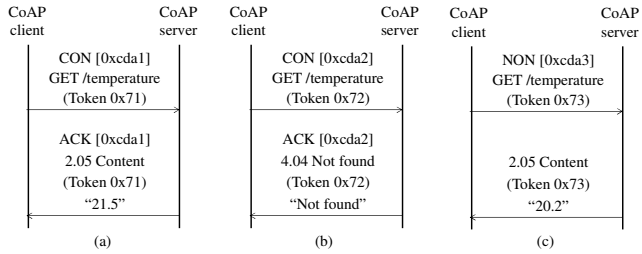

Fig. 7. Two GET requests with Piggybacked responses [6]: (a) successful access to the resource and (b) resource not found (c) request and response carried in non-confirmable message.

H. Security

The security is one of the main problems to be solved in the IoT scenarios [14]. The MQTT protocol was not designed with security in mind and, as many other protocols based on TCP, it uses the Security Socket Layer (SSL) or Transport Layer Security (TLS) for security. When a CONNECT message is sent to the broker, a username/password could be used for authentication. But, as highlighted by Collina et al (2012), the username and password credentials are transmitted without any encryption, given rise to one of the security problems in the protocol. However, an important point is that the MQTT is payload agnostic, so the payload can be encrypted in some level for increasing the security in the communication. For some applications where the transferred information is not sensitive, TLS/SSL may be too computationally expensive and only payload encryption may be enough.

To ensure safety during the exchange of messages among client and server, CoAP uses the DTLS protocol (Datagram Transport Layer Security), which is based on TLS (Transport Layer Security) over UDP instead of TCP. The DTLS has security problems as well as in TLS. One problem is related to achieving DTLS translation when CoAP mapping is used at a proxy for providing end-to-end secure connection. Another issue concerns secure multicast communications that are not yet supported [18].

I. Open source implementations

There are several open source implementations for both protocols, written in different programming languages such that: C, C++, Java, Python, JavaScript, Go, Objective-C and many others[4]. The implementations vary in coverage of protocols specifications and few of them have some kind of limitation. For example, certain C implementations of the MQTT client don't have QoS type 2 implemented because of

[4] For MQTT a reference list of available implementations may be found at https://github.com/mqtt/mqtt.github.io/wiki/libraries and for CoAP at http://coap.technology/impls.html.

the increase in communication overhead, code size and memory consumed. For MQTT-SN, the lack of a well-defined specification, and the need of fitting customized hardware architecture, leads to implementations that are incompatible with each other. Besides presenting a problem, for some applications running over very constrained resources, it is necessary a specific implementation.

Table I summarizes the comparison among the main features of the proposed M2M protocols.

Table I – IOT PROTOCOLS COMPARISON TABLE.

	MQTT	MQTT-SN	CoAP
Network Protocol	TCP/IP	Not specified	UDP
Payload type	Binary	Binary	Binary
Suitable for microcontrollers	Yes	Yes	Yes
Security	SSL/TLS	Not specified	DTLS
Scalability	Simple	Simple	Complex
Network architecture	Broker based (publish/subscribe)	Broker based, client/server, client/client	Client/server (request/response)
Communication pattern	Topic based	Topic based	REST architecture
QoS options	Yes	Yes	Yes
Open Source Availability	Yes	Application Specific	Yes

IV. CONCLUSIONS

This paper provided a qualitative comparison among some important approaches for M2M protocols applied to constrained ICT environments, more specifically the Internet of Things. Each protocol has its own characteristics, which makes them more suitable for a specific situation. The use of TCP as transport protocol limits its usage in more constrained environments. Therefore, more lightweight transport protocols, like 6LoWPAN and ZigBee®, are being employed enabling new possibilities for constrained devices. The message flow control can be configured according to the necessity using the available QoS options in each protocol, which in turn increases data rate and delay. Security is one of the main problems for all approaches and must be reviewed for future versions. In terms of scalability, all proposed protocols can achieve scalability, where the protocols based on publish/subscribed model have simpler implementation.

There are many open source implementations for both protocols written with a diversity of programming languages that cover most application needs.

Finally, MQTT presented better accordance to the presented communication patterns besides that has a lightweight and simple implementation. In the other hand, CoAP can be applied in the context of each thing being a resource that can be accessed through an URL. This feature allows CoAP devices easily adapt to the current web services available.

In summary, the choice depends on application scenario and more than one protocol can be used depending on the requirements of the entire system.

ACKNOWLEDGEMENTS

We would like to thank the support of the Department of Computer Engineer and the Department of Telecommunication Engineer of the Instituto Nacional de Telecomunicações.

REFERENCES

[1] W. Leister and T. Schulz, “Ideas for a Trust Indicator in the Internet of Things,” in *The First International Conference on Smart Systems, Devices and Technologies (SMART 2012)*, 2012, no. c, pp. 31–34.

[2] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, “Key management systems for sensor networks in the context of the Internet of Things,” *Comput. Electr. Eng.*, vol. 37, no. 2, pp. 147–159, Mar. 2011.

[3] C. P. Mayer, “Security and Privacy Challenges in the Internet of Things,” *Challenges*, vol. 17. 2009.

[4] M. Collina, G. Corazza, and A. Vanelli-coralli, “Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST,” in *23rd Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications Introducing*, 2012, pp. 36–41.

[5] International Business Machines Corporation (IBM), “MQTT: Message Queuing Telemetry Transport, version 3.1, protocol specification.” Eurotech, pp. 1–42, 2010.

[6] Z. Shelby, “RFC 7252: The Constrained Application Protocol (CoAP).” pp. 1–112, 2014.

[7] C. Pereira and A. Aguiar, “Towards Efficient Mobile M2M Communications: Survey and Open Challenges,” *Sensors*, vol. 14, no. 10, pp. 19582–19608, 2014.

[8] N. O’Leary, “Paho - Open Source messaging for M2M,” *Eclipse Paho’s MQTT*, 2014. [Online]. Available: http://www.eclipse.org/paho/. [Accessed: 21-Dec-2014].

[9] M. Kovatsch, S. Duquennoy, and A. Dunkels, “A Low-Power CoAP for Contiki,” *2011 IEEE Eighth Int. Conf. Mob. Ad-Hoc Sens. Syst.*, pp. 855–860, Oct. 2011.

[10] T. Rault, A. Bouabdallah, and Y. Challal, “Energy efficiency in wireless sensor networks: A top-down survey,” *Comput. Networks*, vol. 67, pp. 104–122, Jul. 2014.

[11] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, “MQTT-S – A Publish/Subscribe Protocol For Wireless Sensor Networks,” in *3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE ’08)*, 2008, pp. 791–798.

[12] T. Levä, O. Mazhelis, and H. Suomi, “Comparing the cost-efficiency of CoAP and HTTP in Web of Things applications,” *Decis. Support Syst.*, vol. 63, pp. 23–38, Jul. 2014.

[13] C. Bormann, A. P. Castellani, Z. Shelby, U. Bremen, and Z. S. Sensinode, “CoAP: An application protocol for billions of tiny internet nodes,” *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, 2012.

[14] J. Sen, “A Survey on Wireless Sensor Network Security,” *Comput. Networks*, vol. 52, no. 12, p. 24, 2010.

[15] J. H. Kong, L.-M. Ang, and K. P. Seng, “A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments,” *J. Netw. Comput. Appl.*, pp. 1–36, Oct. 2014.

[16] L. Shinho, K. Hyeonwoo, H. Dong-kweon, and J. Hongtaek, “Correlation Analysis of MQTT Loss and Delay According to QoS Level,” *IEEE*, pp. 714–717, 2013.

[17] E. G. Davis, A. Calveras, and I. Demirkol, “Improving packet delivery performance of publish/subscribe protocols in wireless sensor networks.,” *Sensors (Basel).*, vol. 13, no. 1, pp. 648–80, Jan. 2013.

[18] M. Brachmann, O. Garcia-morchon, and M. Kirsche, “Security for practical coap applications: Issues and solution approaches,” *2011 10th GI/ITG KuVS Fachgespraech Sensornetze (FGSN 2011)*, no. Fgsn, pp. 1–4, 2011.